Unit 1: Getting Started with Python and the TI-Innovator™ Hub

Skill Builder 1: Light It Up

In this lesson, you will learn the basics of writing and running a Python program and using the 'light' (the red LED) on the TI-Innovator Hub.

Objectives:

- Create and run a Python program
- Control the light on the TI-Innovator Hub

Teacher Tip: This course introduces students to the TI-Innovator Hub **Hub** (with sketch version 1.5 or higher) and Python programming on the TI-84 Plus CE Python Edition and assumes students have no programming experience. For a more direct introduction to Python programming, see the other Python course on TI-84 Plus CE Python available at education.ti.com > 'TI Codes'.

TI-Python is based on CircuitPython, a variant of Python designed to fit in small microcontrollers. The original CircuitPython implementation has been adapted for use by Texas Instruments, Inc.

In order to use devices on or attached to the TI-Innovator Hub, a Python program must import one or more 'components' (modules) such as **import light** for the red LED in this lesson. For Rover (Units 4, 5, 6) you must **import ti_rover as rv** and, as a result, all Rover statements begin with '**rv.**'. For the TI-RGB Array (Unit 7): **from rgb_arr import *** and then create an instance of the RGB Array class: *var*=**rgb_array()**

The first three units of this course focus on the built-in devices of the hub: **LIGHT**, **COLOR**, **SOUND**, and the **BRIGHTNESS** sensor. The next three units (4, 5, 6) deal with Rover functionality and Unit 7 covers the TI-RGB Array. Texas Instruments has created all the 'ti_' Python modules to support the unique features of the TI-84 Plus CE Python, the TI-Innovator Hub, and the TI-Innovator™ Rover. The TI-Innovator Hub and Rover modules use an Object-Oriented approach to coding, so the commands and the syntax are very different from TI-Basic and even the other Python course.

As a matter of convenience and efficiency variable names are kept to a minimum length whenever possible. Most programming statements can be chosen from the many menus and some functions appear on multiple menus.

We use the **[math]** key as a substitute for the *softkey* **<Fns…> Modul** (keystrokes: **[y=] [leftarrow]**).

Welcome to the world of TI-Innovator Hub programming using Python on your **TI-84 Plus CE Python**. Your calculator must be labeled 'PYTHON' above the screen.

Your first program will operate the red LED on the TI-Innovator Hub circuit board. It is hard to see on the board, but when you turn it on you will know it.

1. To get started with Python programming, press the **[prgm]** key on your calculator and select the **Python App.** The app loads available Python files into the File Manager screen shown to the right. There are three demo files included from the factory but the list can include other files that may have been created. As you make more files, they will be added to this list alphabetically. On the bottom of the screen are five 'soft keys': **<Run>**, **<Edit>**, **<New>**, **<Shell>** and **<Manage>**. Select a 'soft key' using one of the five graphing keys at the top of the keypad. These 'soft keys' will change depending on the current selection.

   Note: the top of your calculator will have the word 'PYTHON' below TI-84 Plus CE. The image shown here comes from the emulator software (TI SmartView™ CE).

Teacher Tip: The entire keypad is remapped to access Python features. See the online e-guide for a more complete explanation of the keypad: https://education.ti.com/html/webhelp/EG_TI84PlusCEPY/EN/index.html

"…available files…" means Python files that are stored in RAM.

2. Start a new program by selecting the **<New>** soft key (press the **[zoom]** key). Since your first program will control the red LED on the TI-Innovator hub, name the program LITE. Your keypad is set to UPPERCASE alpha as the blinking cursor indicates, so just type the letters L-I-T-E and then…

   Note that this screen image (and the rest of the screen images in this course) does not include the top row of keys but only the on-screen 'soft keys'.

Teacher Tip: <Types> options are discussed in a later lesson. **<Esc>** is used to 'back out' of the current screen. **<Ok>** is equivalent to the [enter] key on the keypad.
LIGHT is not a good filename to use because it is already stored as a hidden module.

3. … select the **<Ok>** soft key or press the **[enter]** key. You are now using the Python Editor as indicated on the top of the screen. The five soft keys now contain tools for editing your program:

   <Fns…> contains many programming commands in lots and lots of sub-menus.
   <a A #> is an on-screen 'Character Map' and one-line editor with some useful programming symbols.
   <Tools> contains some useful editing tools, especially 'Undo Clear'.
   <Run> will run your program in the Python **Shell**. Select **<Editor>** from the Shell screen to return to this Editor.
   <Files> takes you back to the File Manager. In the File Manager first point to your file and then select **<Edit>**.

   You can try any of these keys. They contain either an **<Esc>** option or an **<Edit>** or **<Editor>** option.

4. To control the TI-Innovator Hub LED you must start your program with a special TI-developed module called 'light':

   Select **<Fns…>**. On the next screen shown to the right, select the **Modul** menu by pressing the right-arrow key 5 times or the left-arrow key just once. Take a moment to explore the other five sub-menus.

   From the **Modul** menu select the **ti_hub…** menu option as shown using the arrow keys and the **[enter]** key (or press **[6]**).

5. On the next screen, select **Hub Built-in devices…** (since the red LED is built into the TI-Innovator Hub) by pressing **[enter]** or **[1]**.

6. Finally, select **Light** from the **Hub Built-in devices** menu. Here you can either use the **<Import>** soft key or just press **[enter]** or **[2].**

7. All that menu navigation results in the statement

   import light

   pasted as the first line of code in your program. The word '**import**' is blue because it is a Python 'reserved' word. The Editor uses color to highlight keywords, symbols, and literal strings as you will soon see.

   Note ~~Y~~our cursor is on PROGRAM LINE 0002 of the Editor now as indicated in the status line at the top of the screen.

Teacher Tip: It is possible to simply type the statement. turn on alpha-lock by pressing [2nd] [alpha] and type the characters. Use the **[0]** key for the space between the words. Or select the **<a A #>** soft key and begin typing, again using the **[0]** key for a space. Then select **<Paste>** to paste the line into the Editor.

8. For the next statement, press the **[math]** key on the keypad which now directly brings up the **Modul** screen as shown here. But note that there is now a new menu item at the bottom of the list: **Light…**
Select the **Light…** menu item and see…

Teacher Tip: The **[math]** key is a shortcut for **<Fns…> Modul**. This is a perfect example of how the keypad is redefined for the Python experience.
Light… now appears on the **Modul** menu due to the mere presence of **import light** in the code! That is, the code in the Editor actually modifies the menu system! If you delete the **import light** statement (or misspell it) then the **Light…** menu option will disappear!

9. … the three **Light** functions: **on( )**, **off( )** and **blink(freq, time).**
Select **on( ).**

10. Your program now contains two statements.

With your TI-Innovator attached to your calculator, select the **<Run>** soft key to execute the program. Did a red light come on in the Hub?

Teacher Tip: If the TI-Innovator Hub is not attached then an error message will appear: "TI-Innovator not present.". The other information above this message indicates the lines and files that are used in determining this error.

light.on() is a much more subtle statement than it looks. light is a TI-developed Python module and **on( )** is a function in that module. Using the format **import light** requires that 'light' precede any function within that module. This is the power of Python at work: it makes physical programming accessible to even complete beginners and the program is (almost) self-documenting.

11. If your program runs successfully, you will see the screen to the right and the red LED lights up on the TI-Innovator hub. ***Congratulations!*** You have written your first TI-Innovator program using Python.

    The screen now displays the **Python Shell** as indicated at the top of the screen. The messages on the screen indicate the last actions performed. The two lines beginning with **>>>** # (the *octothorpe*) are comments. The line >>> **from LITE import *** is the command that executes the code in your LITE program. The last line is the Shell prompt **>>>** and the cursor | on this line is waiting for you to enter a Shell command because your program is done.

But note that the red LED on the hub is still lit. How can you turn it off?

Teacher Tip: The Python Shell is the workspace where Python programs are executed. It is similar to (but not the same as) the calculator HOME screen in that you can enter valid Python expressions that can be processed immediately. You can also use the Shell to test statements and short blocks of code.

The three workspaces in the Python App are Editor, Shell, and File Manager.

Editor: edit a Python file
Shell: run a Python program; test Python expressions and statements
File Manager: copy, rename and delete Python files.

To quit using the Python App press **[quit]** (**[2nd] [mode]**) at any time except when a program is running.
To 'break' a program press the **[on]** key.

12. Return to your program code by selecting **<Editor>**. Your cursor should be blinking at the end of the **light.on( )** line 0002. Press **[enter]** to move the cursor to the next line. Add a statement to turn the light off. Try it yourself before going to the next step.

13. Did you use the **light.off( )** function? It's found on **[math] Light…** just below **light.on( )**.

    <Run> the program again and watch the TI-Innovator hub carefully. You should see the red LED quickly flash. To repeatedly run the program simply press the **<Editor> <Run>** soft key since it toggles between the Editor and the Run operation (Shell).

14. To control the timing of the LED (how long it stays on before turning off) you will add a **sleep( )** function …

15. Back in the **<Editor>** add two blank lines to your code by pressing **[enter]** at the beginning of each **light** statement. Now, with your cursor on the line below the **import light** statement…

16. Add the statement

    from time import *

    found on **[math] time…**

17. Now, between **light.on** and **light.off**, add the **sleep( )** function also found on **[math] time.** (Simply use the arrows on keypad to navigate up and down lines in your code.)

Inside the **sleep( )**, type a number of seconds inside the parenthesis. This statement will thus pause processing for that many seconds before proceeding to the next statement so that the LED stays on for some seconds before turning off.

Run the program now and watch the LED.
After the **light.off()** statement, give the **light.blink( )** command a try (note that it takes two arguments separated by a comma (provided for you).

See if you can figure out what the two numbers are used for! Experiment a little, changing the numbers, and watching what happens..

Teacher Tip: : light.blink(freq, time) is displayed on the menu as a tip but the blue text is not pasted into the code. It's up to the programmer to remember the tips:

freq is the blinks-per-second and
time is the total number of seconds to blink.